Embedded Applications Journal

A PUBLICATION OF INTEL CORPORATION THIRD QUARTER, 1993

What's Inside

*Project*BUILDER196, A New Application Developer's Kit

Will Schreiber
Senior Program Manager
Intel Corporation
Article ID #0501

*Project*BUILDER196 is a low-cost ($196.00) kit that answers the Design Engineer's question, "How well can the 8XC196 embedded controller handle my application?" This turnkey kit includes *Model*BUILDER, Demo ASM-96, a 196KD target board, a retargetable symbolic debug monitor, and more. Using *Model*BUILDER, a system-level application modeling and performance analysis tool, you can quickly evaluate the 8XC196KB, 8XC196KC, and 8XC196KD embedded controllers without writing a single line of code.

Use your mouse to select from nine precoded performance templates in a model control loop. Then change the execution period of each template to best represent your application load. The resulting CPU loading percentage is graphically displayed on the color-coded bar chart and in tabular form.

Next, perform "what if" hardware design analysis with pull-down menus for changing memory wait states, address bus width, and CPU frequency settings. This invaluable tool helps you find the optimum price/performance 8XC196 design strategy within minutes.

Profile Your Application Performance

With the *Project*BUILDER kit, you can load one module of your own code (ASM-96 or C-96) into *Model*BUILDER and combine it with the performance templates. The software times your code and factors the result into the "what if" hardware design analysis. You can profile your application's CPU load with a worst-case ±5% confidence, and you can save up to five models for quick comparisons.

Turnkey Application Environment

*Project*BUILDER provides a complete turnkey Windows*-based application development environment. In addition to *Model*BUILDER, the kit comes with a Demo ASM-96 and a retargetable symbolic debug monitor for low-level interrogation of the 20 MHz 196KD Target Board. You can download your code at 57.6 Kbaud. Change one jumper and the board will operate as a stand-alone system. Debug and interrogate registers or memory in real time while the CPU is running. Set up to 16 breakpoints in your code, assemble or disassemble, or just step through execution line by line. The on-chip monitor kernel is less than 750 bytes in size. The source code is annotated to show you how to retarget the monitor to your prototype hardware. It is configurable for the 8XC196KB, 8XC196KC, or 8XC196KD at any frequency.

And More!

The *Project*BUILDER kit includes the new *Ap*BUILDER version 1.2 with hypertext technical documentation. *Ap*BUILDER now generates ASM-96 & C-96 peripheral code as functions that use programming macros to speed relocation. OrCAD** board schematics and library of 114 devices, source code for all performance templates, and the monitor kernel are also in the kit. Customers who register

Continued on page 2

From the Managing Editor

Have you tried the BBS?

The Intel Applications Bulletin Board is up and waiting for your call. FaxBACK® service document #2009 explains the main features of the BBS and helps first-time users to log on. The BBS supports all of Intel's components — microprocessors, LANs, ExCA™ cards, Indeo™ video technology, and embedded products — with formats of up to 14.4 Kbaud with no parity, eight data bits, and one stop bit.

First-time users have file download privileges. Although initially limited to 2 Mbytes, users will be upgraded after the first 24 business hours to unlimited usage. There are 16 modem lines connected to the server. Two of these lines are dedicated to 2400 baud, and the rest support up to 14.4 Kbaud v.32, v.42. We monitor the call load and add new lines as they're needed, so there should be no busy signals.

You can download such files as 16/32 bit math routines for the 8051; a routine for software emulation of I2C for the 8051; evaluation board schematics (in OrCAD* format) for the 8051, the 8096, the i960® component, and many others; *Ap*BUILDER v1.2 software; bug lists and workarounds; 186 software drivers for bootblock Flash; and 186 software routines for interfacing to an 82077AA floppy disk controller chip. Just to name a few. So check it out today.

Article Numbers and Feedback

We need your feedback in order to improve. In the last issue, we changed the feedback response section to make it easier for you to communicate the quality and usefulness of each article in the Journal. Each article has a number listed below the author's name. You simply record the article number and check off the quality and usefulness indicators. Then fax it to us. It's that simple.

Steven M. McIntyre
Embedded Applications Manager
Embedded Microcomputer Division

*OrCAD is a registered trademark of OrCAD Inc.

More Copies

Need more copies of this issue of the *Embedded Applications Journal*? In the U.S. and Canada, call Intel Literature at 800-468-8118 and order #241294-005. In Europe and other international locations, please contact your local Intel sales office or distributor for additional copies.

Intel Support Numbers

Customer Support	(U.S. and Canada)	800-628-8686
Customer Training	(U.S. and Canada)	800-234-8806
Literature Fulfillment	(U.S. and Canada)	800-468-8118
FaxBACK® Service	(U.S. and Canada)	800-628-2283
FaxBACK® Service	(Worldwide)	916-356-3105
Applications BBS	(Worldwide)	916-356-3600
up to 14.4 Kbaud lines		916-356-3605
dedicated 2400-baud lines		916-356-7209

*Project*Builder196, A New Application Developer's Kit

Continued from page 1

*Project*BUILDER196 will receive a Demo C-96 compiler in Q3 '93.

For a **free** demo diskette of *Model*BUILDER, please call Intel Literature at 800-468-8118 and ask for #272329. In Europe and other international locations, please contact your local Intel sales office or distributor.

*Windows is a trademark of Microsoft Corporation. **OrCAD is a registered trademark of OrCAD Inc.

Using Fuzzy Logic With Microcontrollers: An Overview

Hong Soo Ng
Applications Engineer
Intel Corporation
Article ID #0502

Introduction

Fuzzy logic is a technology based on fuzzy set theory developed by Professor Lofti Zadeh in 1965. Over the last few years, it has been finding a rapidly increasing number of applications in consumer electronics and embedded control systems. Fuzzy logic provides a new way of solving problems using a human-like reasoning process rather than an exact mathematical modeling of the system. As a result, in many cases, it reduces the product development cycle and produces systems that are more robust and have more features. In addition, most fuzzy logic applications can be easily implemented using standard microcontrollers.

Fuzzy Logic vs. Conventional Technology

Solving control problems using conventional methods or algorithms requires complete knowledge of the system. Many times, the system is nonlinear or so complex that it either requires a high-order mathematical equation to model or yields no clear mathematical model. This makes the problem very difficult or even impossible to solve. Unlike conventional technology, fuzzy logic uses a rule-based system architecture that allows the system to be described by imprecise or human-like statements and variables.

In general, using fuzzy logic has the advantages of lowering product development cost; reducing time to market; and improving product performance, reliability and robustness. Fuzzy logic is a better solution than conventional technology when systems are too complex to model accurately, when they have operational nonlinearities, or when they have uncertainties in either their inputs or definition. However, fuzzy logic is not the solution to all problems. If a system can easily be accurately modeled and optimally solved with conventional technology, there is no reason to turn to fuzzy logic.

Fuzzy Logic Basics

Fuzzy logic involves three primary processes: fuzzification, fuzzy rule inference, and defuzzification (Figure 1). During fuzzification, crisp input values are translated into a degree of membership function (a fractional value from 0 to 1) in a number of fuzzy sets, using predefined input membership functions. The fuzzy inputs are then evaluated using predetermined linguistic rules to produce fuzzy outputs. Finally, all fuzzy outputs are combined and translat-

Continued on page 4

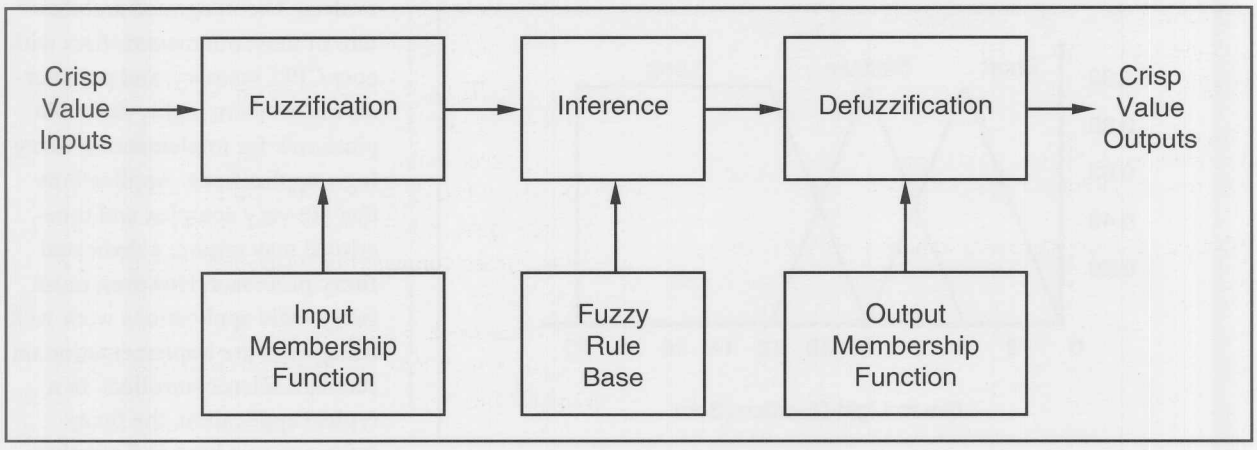

Figure 1. Fuzzy Logic Processes

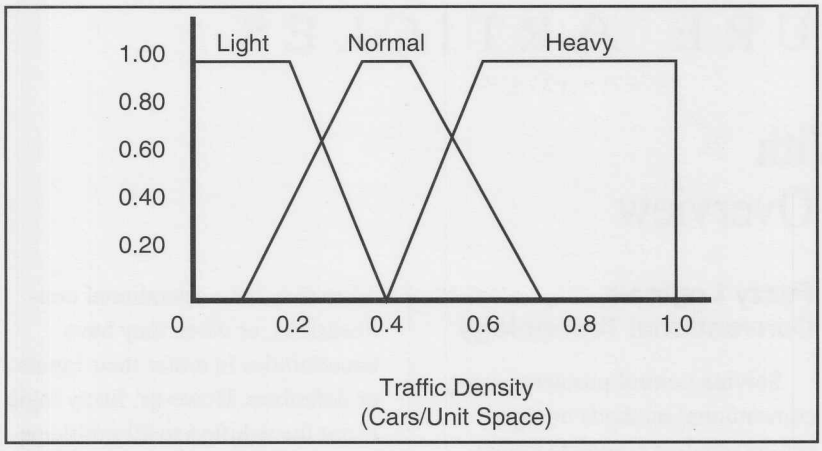

Figure 2. Input Membership Function for Traffic Density

Continued from page 3

ed into crisp and executable output values, using output membership functions during the defuzzification process.

Consider a simplified traffic control problem. The input to the system is traffic density and the output is the green light duration. Using intuition, we can easily come up with a few rules to the system:

Rule 1: If traffic is light, then green light is short.

Rule 2: If traffic is normal, then green light is medium long.

Rule 3: If traffic is heavy, then green light is long.

We notice that terms such as light, heavy, normal, short and long are all linguistic terms from everyday language, without real precision, and thus fuzzy. Unlike classical set theory, in which a crisp value can be only either a member or not a member of a given set, fuzzy logic allows a value to have degrees of membership in different sets. In this example, the input membership function for traffic density and the output membership function for green light duration are determined as in Figure 2 and Figure 3.

Assuming that a crisp input value of 0.3 comes in to the system, after fuzzification, the value will have a membership value of 0.5 to "light" and 0.8 to "normal". In other words, the traffic density is 50% light and 80% normal. This will fire Rule 1 and Rule 2 simultaneously. In the fuzzy rule inference process, the degree to which a rule's condition is true will determine the degree to which the resulting action is taken.

This is done by activating the area of output membership functions to the degree to which the rule's condition is true. In this case, it means that the action to turn on the green light for a medium long time will be executed with a greater strength than the action to turn on the green light for a short time. These two actions will be combined and then defuzzified to produce a crisp value output that can be executed. The most frequently used defuzzification method is called Center-of-Area (CoA), in which the center of area location to all applicable output membership functions is calculated to produce the crisp output value. The process is illustrated in Figure 4.

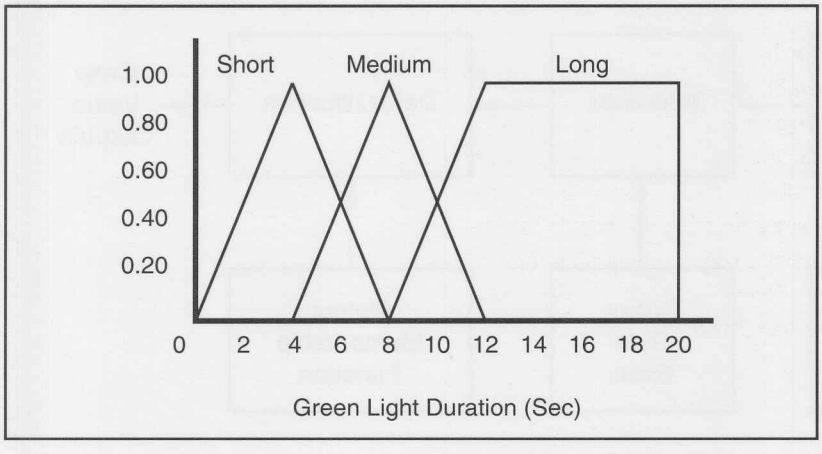

Figure 3. Output Membership Function for Green Light Duration

Using Fuzzy Logic with Microcontrollers

Most fuzzy logic applications are currently implemented in software using standard microcontrollers. The integrated architecture of most microcontrollers with core CPU, memory, and peripherals makes them highly desirable platforms for implementing fuzzy logic applications. Applications that are very complex and time-critical may require a dedicated fuzzy processor. However, most fuzzy logic applications work well using software implementation on standard microcontrollers. In a typical application, the fuzzy inference rule base and member-

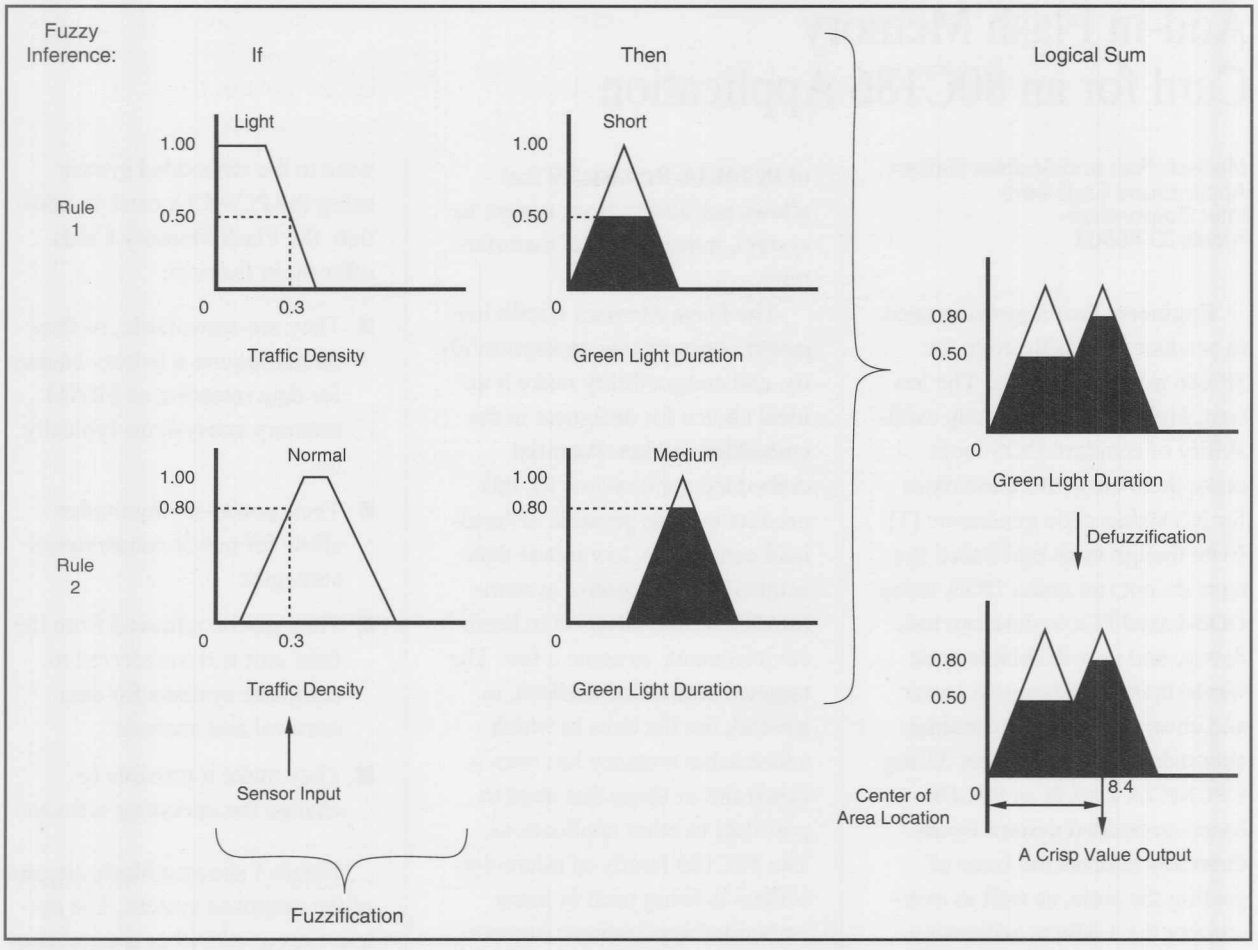

Figure 4. Fuzzy Rule Inference Process

ship function information can be stored as a look-up table in the memory available onboard most microcontrollers. Fuzzification, fuzzy rule inference, and defuzzification processes are then implemented in software using multiply and accumulate functions.

A number of fuzzy logic development tools and products are currently available in the market. These tools greatly simplify the development work by providing graphical tools for fuzzy logic design, optimization, and verification. Inform Software Corp., a pioneer of fuzzy logic applications in Europe, is currently working closely with Intel in providing fuzzy logic development tools for Intel's embedded microcontrollers and microprocessors. Inform's *fuzzy*TECH software provides a window-based, integrated graphical environment for fuzzy logic application development, debugging, simulation, and verification. It can also generate C code as well as highly optimized assembly code for Intel's MCS®-96 and MCS®-51 embedded microcontrollers. Together with this user-friendly development environment, Intel's embedded architecture provides an easy and efficient way to implement fuzzy logic applications.

**fuzzy*TECH is a trademark of Inform Software Corp.

Add-in Flash Memory Card for an 80C186 Application

Mahesh Rao and Andrew Gafken
Applications Engineers
Intel Corporation
Article ID #0503

Engineers face a growing need to produce ROMable code for 80X86 microprocessors. The low cost, high quality, and ready availability of standard DOS tools make them attractive candidates for ROMable-code generators [1]. Even though most embedded systems do not run under DOS, using DOS-based PCs to develop, test, debug, and port ROMable code would make development easier and cheaper and could considerably reduce time to market. Using a PCMCIA card in an 80C186-based embedded system would definitely address the issue of porting the code, as well as overcoming the 1 Mbyte addressing constraint of the 80C186 [2]. This article discusses a simple system implementation of a PCMCIA card for an 80C186-based board.

PCMCIA–80C186 Interface

Intel's Series 2 Flash Memory Cards facilitate high-performance disk-emulation in mobile PCs and dedicated equipment [3]. Series 2 Flash Memory Cards conform to the Personal Computer Memory Card International Association (PCMCIA 2.0)/Japanese Electronics Industry Development Association (JEIDA 4.1) 68-pin standard, providing electrical and physical compatibility. In addition, the Series 2 Flash Memory Card is compatible with Intel's Exchangeable Card Architecture (ExCA™), an open hardware and software system implementation of PCMCIA Release 2.0 that allows portability from system to system, independent of manufacturer.

The Flash Memory Card's low power consumption, transportability, and compatibility make it an ideal choice for designers in the embedded market. Potential embedded applications for this product include portable or handheld controllers, low power data acquisition and control systems (DAQMs), and DAQMs in harsh environments, to name a few. The targeted embedded markets, in general, are the ones in which addressable memory has been a constraint or those that need to port data to other applications. The 80C186 family of microcontrollers is being used in many embedded applications; however, the 80C186 can directly address a maximum space of 1 Mbyte, which limits its usage in some applications.

System Details

This article describes an add-in board that would increase the addressable memory space of both future and existing 80C186 platforms, with minor modifications in some cases. Since the add-in would be based on Series 2 Flash Memory Cards, the interface would meet the PCMCIA standard. Meeting the PCMCIA standard would inherently increase the 80C186 memory space addressability from 1 Mbyte to 64 Mbytes. This add-in board would allow designers to develop, test, and debug the software on a DOS-based PC, then download the software to the embedded system using the PCMCIA card. In addition, the Flash Memory Cards offer these features:

- They are nonvolatile, so they do not require a battery backup for data retention as SRAM memory subsystems typically do.
- Their power-saving modes allow for power conservation strategies.
- They can be removed from the field unit and transferred to computer systems for data retrieval and analysis.
- They make it possible to change the operating software.

Figure 1 shows a block diagram of the proposed system. The system uses an 82510 as a serial controller to communicate with the outside world. Its main components include the following:

- 128 Kbytes of SRAM
- 16 Kbytes of program memory (EPROM)
- 128 Kbytes of Flash
- a PCMCIA interface with addressing capabilities for cards up to 64 Mbytes
- an 82510 serial controller for RS-232C-based communication capabilities
- an 82C55 Programmable Peripheral Interface Adapter (PIA)

To address the 64 Mbytes of the address space, the 80C186 system performs the following steps:

1. Sets up the appropriate page

address using the 82C55. Each page is 64 Kbytes in length, and the page can be set up by writing to the port.

2. Performs the Memory Read or Memory Write operation to the PCMCIA page address space of the 80C186 (physical address 30000H).
3. Rewrites the page address space of the 82C55 to indicate Page 0.

The above three-step procedure has appropriate messages to indicate the actions taken. This is accomplished using the 82510 serial controller.

PCIC Use

Intel's PC Card Interface Controller (PCIC) is the ExCA interface solution for the notebook PCs [4]. The 82365SL allows PC manufacturers to design their systems to provide the PC user with a wide range of connectivity options (such as Modem, Twisted Pair Ethernet) as well as eliminating rotating electromechanical media (by using the Intel Flash memory card). The 82365SL was not used in this design, for the following reasons:

- The scope of the project was to interface the Flash memory cards to the 80C186-based system design, and cost was one of the major considerations. The cost of the 82365SL is greater than that of the 82C55.
- This design would not use the 82365SL's full functionality and capabilities, so the increased system cost would not be justified.
- The complexity of the design using the 82365SL would be greater, in both hardware and software, than that of the proposed design.

PIA Use

This design suggests and implements a solution that is both cost-effective and easy to imple-

Continued on page 8

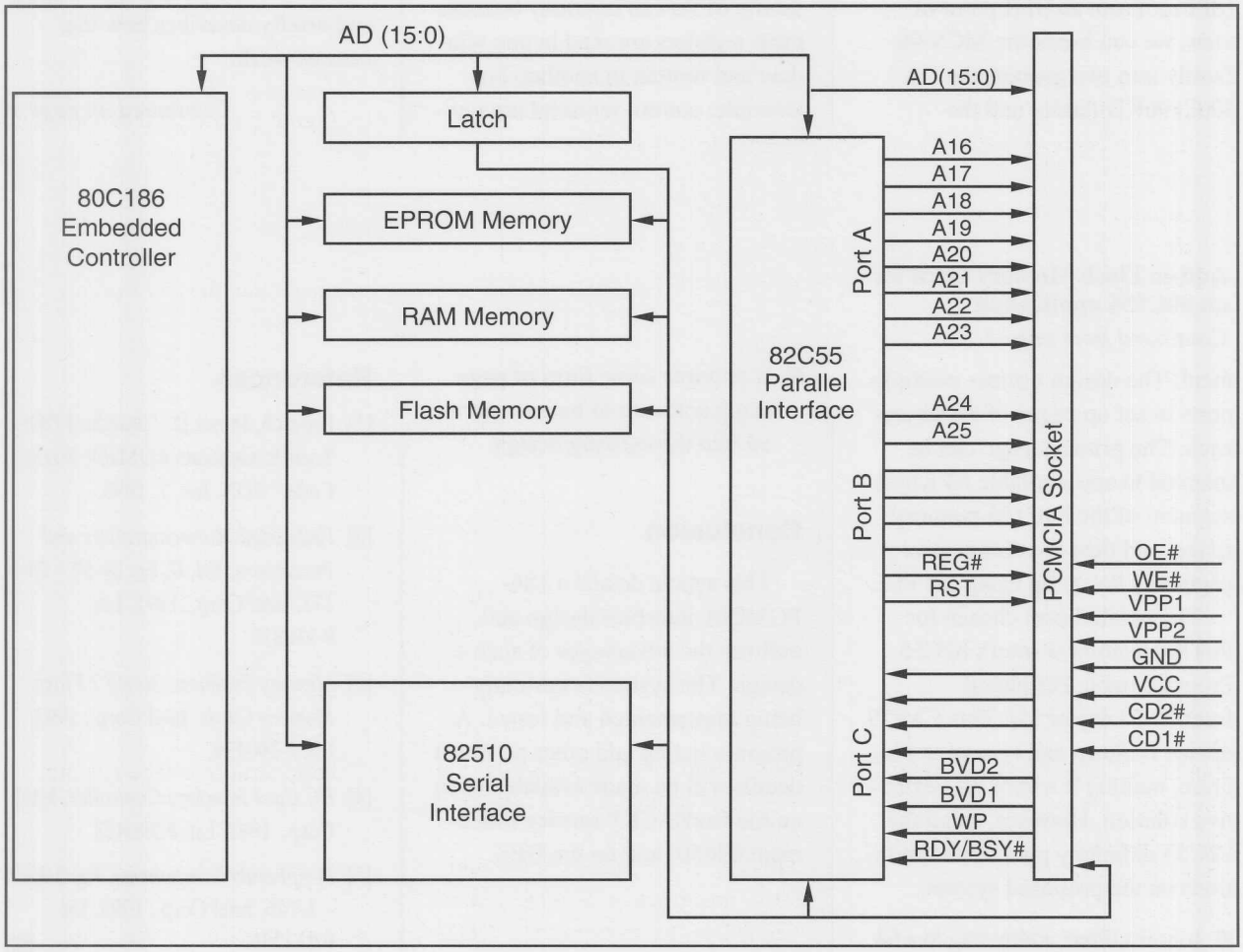

Figure 1. Block Diagram

Macros for Accessing Windowed SFRs on the MCS®-96 Family

Robin Manelis
Technical Marketing Engineer
Intel Corporation
Article ID #0504

The 8XC196KD family macros are available from the BBS (File MACROS.ZIP). The macro "include" files (80c196kd.inc and 80C196kd.h) are also provided with ApBUILDER Version 1.2. ApBUILDER-generated peripheral initialization code uses these macros.

This article provides an overview of macros designed to help you program the MCS®-96 family special function registers (SFRs). From an SFR point of view, we can break the MCS-96 family into two groups: the 8XC196KD family and the 8XC196KR family. The 8XC196KD family includes the devices that require horizontal windowing to access SFRs (80C194, 80C198, 80C196KB, 80C196KC, and 80C196KD). The 8XC196KR family includes the devices that require vertical windowing to access SFRs (80C196KR, 80C196KQ, 80C196JR, 80C196JQ, and 80C196KT).

Accessing 8XC196KD Family SFRs

Accessing the 8XC196KD family SFRs can be tricky because most registers are read in one window and written in another. For example, control registers are typically written in window 0 and read in window 15, while status registers are typically written in window 15 and read in window 0. The tricky part is remembering each register's read and write window. Macros are now available in iC-96 and ASM-96 that make programming the 8XC196KD family SFRs easier. These macros perform such tedious tasks as determining the access window, determining the register size, and changing the window for you.

This article demonstrates how using the 8XC196KD family macros can simplify programming and briefly describes how the macros work.

Continued on page 9

Add-in Flash Memory Card for an 80C186 application

Continued from page 7

ment. The design utilizes multiple ports to set up pages of 64 Kbytes each. The primary page can be mapped to any available 64 Kbyte segment of the 80C186 memory space, and then the consecutive pages can be set up using the PIA.

The parallel port chosen for this application is Intel's 82C55 Programmable Peripheral Interface Adapter [5]. This CMOS device is cheap and is easy to program, making it a very cost-effective solution. However, using the 82C55 definitely puts some limitations on the proposed system:

- It limits direct addressing to a 64 Kbyte block size.
- It requires some form of page setup software to be incorporated into the existing design.

Conclusion

This article details a 186-PCMCIA interface design and outlines the advantages of such a design. The system is currently being implemented and tested. A program listing and other pertinent details will be made available soon on the FaxBACK® service (document #2650) and on the BBS.

References

[1] Broesch, James D., "Standard DOS Tools to Generate ROMable 80x86 Code," *EDN*, Jan. 7, 1993.

[2] *Embedded Microcontrollers and Processors, Vol. II*, Pg. 24-59 – 24-173, Intel Corp., 1993, Lit. # 210830.

[3] *Memory Products, Series 2 Flash Memory Cards*, Intel Corp., 1993, Lit. # 290434.

[4] *PC Card Interface Controller*, Intel Corp., 1993, Lit. # 290423.

[5] *Peripherals Components*, Pg. 3-124 – 3-146, Intel Corp., 1993, Lit # 231256.

Continued from page 8

8XC196KD Family Macro Examples

Table 1 shows how using macros can simplify reading and writing SFRs and setting and clearing SFR bits. All the macros determine the SFR's read window, write window, and size from the SFR name. The code generated by the _ReadSFR and the _WriteSFR macros first loads the Window Select Register (WSR) with the desired window (the SFR's read or write window) and then performs the appropriate operation. The _SetSFR_bit and _ClrSFR_bit macros determine a mask value from the number of the bit to be set or cleared. The code generated by these macros loads the WSR with the SFR's read window, sets or clears the desired bit, loads the WSR with the SFR's write window, and loads the new value into the SFR.

When changing the value of the WSR, the macros do not corrupt the Hold Enable bit (WSR.7). Table 1 shows the instructions generated for byte registers. When the macro call contains a word register, the appropriate instructions are generated (e.g., "ld," "and," "or" instead of "ldb," "andb," "orb").

Two additional macros for the 8XC196KD family devices are _OrSFR and _AndSFR. These macros are similar to the _SetSFR_bit and _ClrSFR_bit macros. The _SetSFR_bit and _ClrSFR_bit macros allow you to set and clear individual bits, while the _OrSFR and _AndSFR macros allow you to set and clear several bits within the same SFR.

How the 8XC196KD Family Macros Work

In addition to the macros, the "include" files contain definitions to support the macros. Each SFR has its address, read window, write window, and size associated with its name. Figure 1 shows a portion of the 80c196kd.inc "include" file.

A macro call passes the name of the SFR to the macro, which then appends "_w" to the name. The macro determines the register's size, read window, and write window from the appropriate SFR_w bits. For example, the ioc0_w symbol defines the read

Continued on page 10

Table 1. Code Generated When Using 8XC196KD Family Macro Calls

Function	Language	Macro call	Code generated by the macro
Reading an SFR	ASM	_ReadSFR destination, SFR	ldb wsr, #ReadWin + _HOLDEN ldb destination, SFR
	C	destination = _ReadSFR (SFR);	wsr = Readwin + _HOLDEN; destination = SFR;
Writing an SFR	ASM	_WriteSFR SFR, #Value	ldb wsr, #WriteWin + _HOLDEN ldb SFR, Value
	C	_WriteSFR (SFR, Value);	wsr = WriteWin + _HOLDEN; SFR = Value;

Setting an SFR bit	ASM	_SetSFR_bit SFR, BitNumber	ldb wsr, #ReadWin + _HOLDEN
			ldb tmpreg, SFR
			orb tmpreg, #MaskValue
			ldb wsr, #WriteWin + _HOLDEN
			ldb SFR, tmpreg

	C	_SetSFR_bit (SFR, BitNumber);	wsr = ReadWin + _HOLDEN;
			tmpreg = SFR || MaskValue;
			wsr = WriteWin + _HOLDEN;
			SFR = tmpreg;

Clearing an SFR bit	ASM	_ClrSFR_bit SFR, BitNumber	ldb wsr, #ReadWin + _HOLDEN andb tmpreg, SFR, #MaskValue ldb wsr, #WriteWin + _HOLDEN ldbSFR, tmpreg

	C	_ClrSFR_bit (SFR, BitNumber);	wsr = ReadWin + _HOLDEN;
			tmpreg = SFR && MaskValue;
			wsr = WriteWin + _HOLDEN;
			SFR = tmpreg;

```
; 80c196kd.inc

; Use the _HOLDEN constant to fix the value of the HLDEN bit (wsr.7). Use 80h to set this
bit and 0 to clear the bit.

 _HOLDEN    set         80h

; Bits 4-7 are set to the read window values, bits 0-3 are set to the write window values,
; and bit 8 is set to a one if the register is a word and left as zero if the register is
a byte.

READ_WIN0           set       000h
READ_WIN1           set       010h
READ_WIN15          set       0f0h
WRITE_WIN0          set       000h
WRITE_WIN1          set       001h
WRITE_WIN15         set       00fh
READ_WRITE_ANY      set       055h
WORD_OP             set       100h

; Each special function register is associated with its address.

zero_reg            equ       00h:word
ad_command          equ       02h:byte
ad_result           equ       02h:word
ioc0                equ       15h:byte
ptssel              equ       04h:word

; Each special function register is associated with a value that contains
; its read window in bits 4-7, its write window in bits 0-3, and a register size indicator
in bit 8.

 zero_reg_w         set       READ_WRITE_ANY + WORD_OP
 ad_command_w       set       READ_WIN15 + WRITE_WIN0
 ad_result_w        set       READ_WIN0 + WRITE_WIN15 + WORD_OP
 ioc0_w             set       READ_WIN15 + WRITE_WIN0
 ptssel_w           set       READ_WIN1 + WRITE_WIN1 + WORD_OP
```

Figure 1. 80C196KD.inc Excerpt

Continued from page 9

window (in bits 4–7), the write window (in bits 0–3) and the register size (in bit 8) for the ioc0 register. From the ptssel_w symbol, you can see that ptssel is a word register that is read and written in window 1.

Accessing 8XC196KR Family SFRs

In a similar manner, you could use macros to make programming the 8XC196KR family SFRs easier. Accessing the 8XC196KR family SFRs can be tricky because most registers are located outside the lower register file. Registers outside the lower register file require vertical windowing in order to be accessed using register direct addressing. Vertical windowing allows you to map sections of memory into the top of the lower register file in 32-, 64-, or 128-byte increments known as windows. To directly address an SFR through vertical windowing, you need to know the window size, the window number, and the address of the memory location relative to the base address of the window. You could use macros to perform such tedious tasks as determining the window number and the relative address.

ZapCode

Renee LaGrow
Business Support Systems
Intel Corporation
Article ID #0505

ZapCode is a breakthrough in the ROM code submittal process. The design positions our customers and Intel to save time and money. It gets information to the customer in minutes, a process that previously took days. This is a standardized process with real-time quality checks, and it runs 24 hours a day. It can reduce the ROM lead-time by cutting down administrative throughput time. This is also a breakthrough in the way the industry does business, and it promotes Intel as a World Class supplier.

Table 1. Before and After Zap Code

Before ZapCode	After ZapCode
24-hour turnaround to transfer, verify, and return customer data	Less than 10 minutes turnaround
ROM processing "open" 8 hours	ZapCode "open" 24 hours
Multiple media types and shipping methods	Standardized process
Incompatible formats sent to ROM processing	Real-time quality checks before code is sent to Intel
Manual ROM processing systems	Automated system
Faxes, FedXs, lots of folks	Electronic files, few folks

ZapCode is an online, PC-based electronic transmittal service that simplifies and streamlines the transfer of ROM media. It converts the customer's code to Intel's standard hex format. It automatically verifies the code, assigns a ROM code number, and sends the code back with the customer verification letter.

Approximately 80 accounts, including AMO, Europe, SAO, Distribution, and APAC customers, are currently using ZapCode for Commercial and Automotive code transmission. ZapCode has a toll-free telephone number that allows U.S. customers to send their code to Intel at no cost.

Getting Started

A ZapCode starter kit includes the User's Guide, ZapCode diskettes, a customer presentation (on request), and an introductory customer letter. The ZapCode starter kit can be obtained by contacting ROM Processing at (602) 554-8618.

The diskettes contain ZAPPREP (ZapCode Preparation program) software. ZAPPREP is a copy of Intel's hex file translation software and the access to the ZapCode system. ZAPPREP allows the customer to convert hex files into Intel's standard hex format before sending it through ZapCode. ZAPPREP is an augmented version of IPPS (Intel PROM Programming Software), using the same algorithms. Intel ROM Processing uses ZAPPREP for converting codes to Intel's standard hex format; it is also used for mask generation. ZAPPREP has automated the conversion process to make the process much easier to use. Our customers use a PC and get all their information by modem.

The following is the required customer PC setup.

- IBM AT-compatible personal computer
- Hayes-compatible modem (2400 baud recommended)
- analog telephone line (tone dial)
- printer
- DOS version 3.3 or greater
- PROCOMM PLUS* software, version 2.0

Continued on page 12

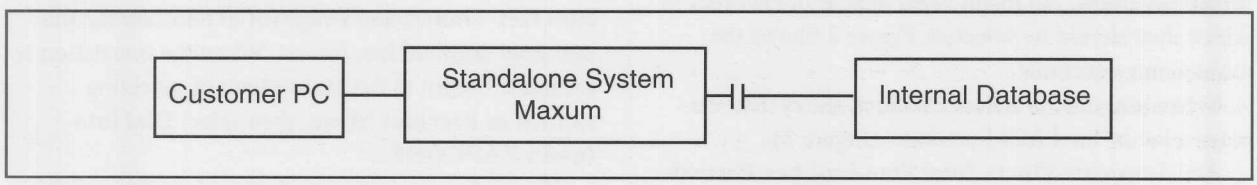

Figure 1. The ZapCode Database Interface Diagram

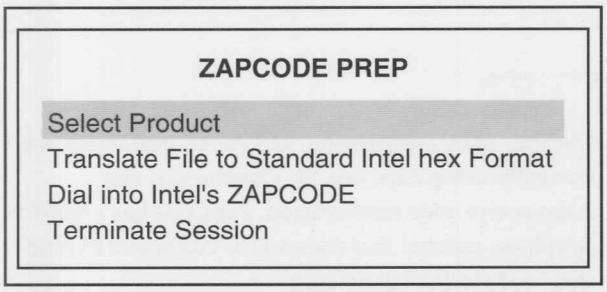

Figure 2. ZapCode Prep Main Menu

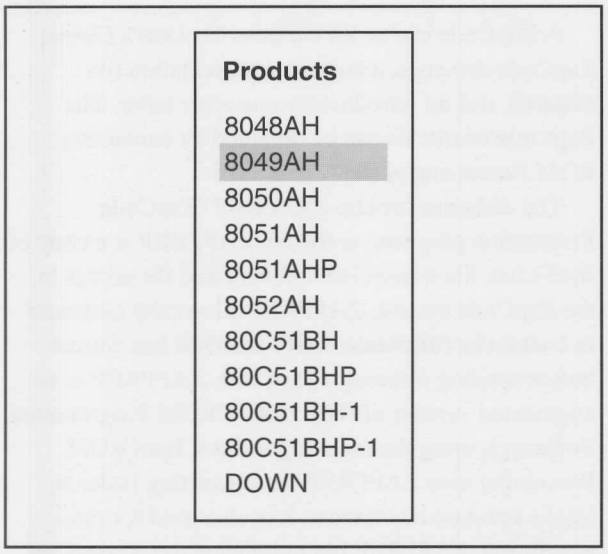

Figure 3. Intel ROM Product List

Continued from page 11

*PROCOMM PLUS is a standard communications package available at computer stores. Its estimated cost in the U.S. is under $100.

Figure 1 illustrates the interface between the PC and the internal database. All the information from the customer is automatically input to the internal database. The information is updated and ported to the stand-alone system. This action takes place at certain times of the day for security purposes.

Using ZAPPREP and ZapCode

ZAPPREP and ZapCode are easy-to-use, menu-driven programs. All menu items appear in the order in which they should be selected. Figure 2 Shows the main menu selections.

When you choose **Select Product**, the system displays a list of Intel ROM products (Figure 3).

The **Translate File to Intel Standard hex Format** selection allows you to translate an ASCII hex or object hex file into Intel's standard hex format (Figure 4).

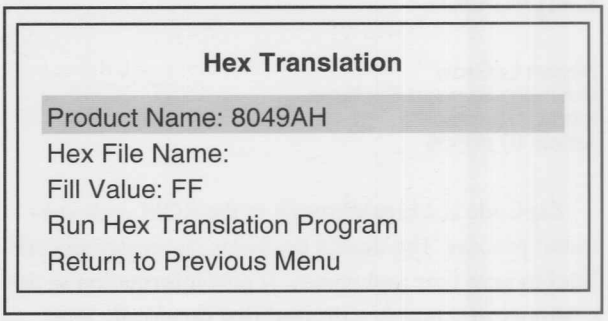

Figure 4. Hex Translation Menu

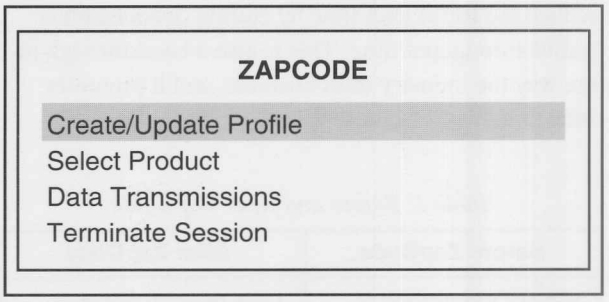

Figure 5. ZapCode Main Menuu

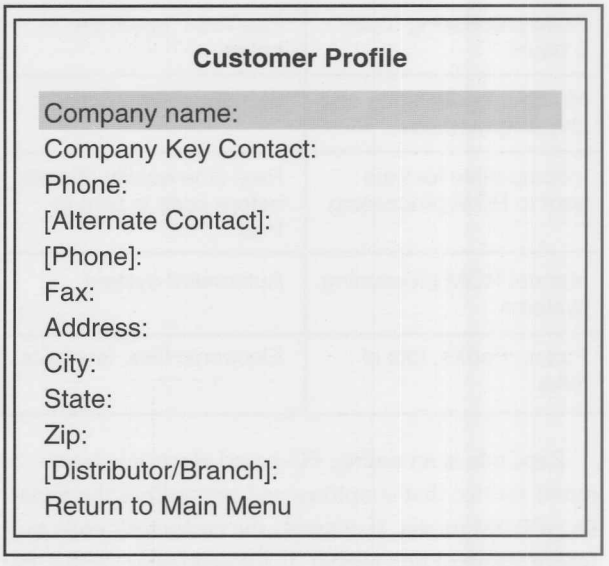

Figure 6. Customer Profile Menu

Enter the hex file name and fill value, then select **Run Hex Translation Program** to translate the file into Intel standard hex format. When the translation is complete, return to the Main Menu by selecting **Return to Previous Menu**, then select **Dial into Intel's ZAPCODE**.

The necessary digits are displayed on the screen.

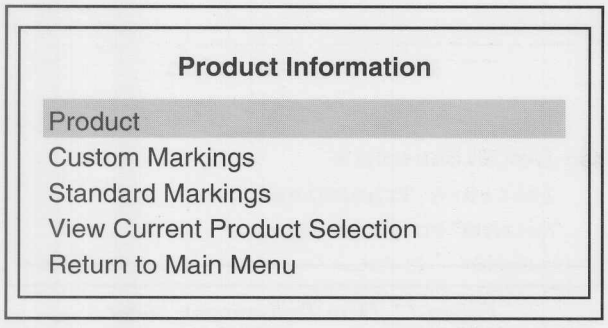

Figure 7. Product Information Menu

The next step is to type in the username and password. At this point, you see the ZAPCODE menu at Intel (Figure 5).

Move down through the menu items listed to transmit the code to Intel. The **Create/Update Profile** selection displays the Customer Profile Menu (Figure 6).

The first time you use ZapCode, you must enter the profile information. Most fields are required information. Those that are not required are marked in parentheses. Once you have set the profile, there is no need to return to this screen on subsequent code submissions unless there are changes to the required information.

The **Select Product** option displays the Product Information Menu (Figure 7).

Selecting **Product** returns the Products, Packages, and Grades menu (Figure 8).

1. Select the Intel product. It must be the same product selected in ZAPPREP.

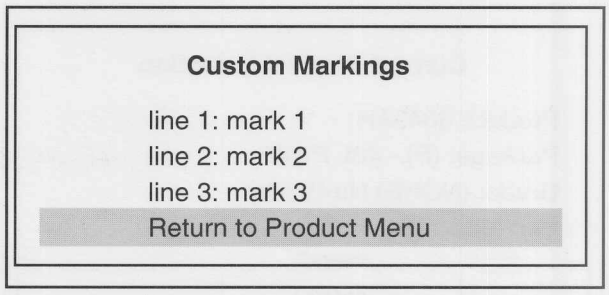

Figure 9. Custom Markings Menu

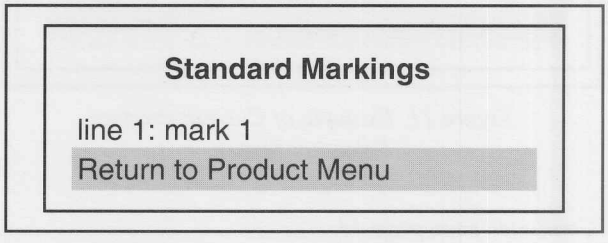

Figure 10. Standard Markings Menu

2. Select the desired package.
3. Select the desired grade.

When you have completed the product, package, and grade selections, the system will automatically return to the "Product Information" menu. The next step is to select either **Custom Markings** or **Standard Markings** to access a product marking menu (Figures 9 and 10).

Continued on page 14

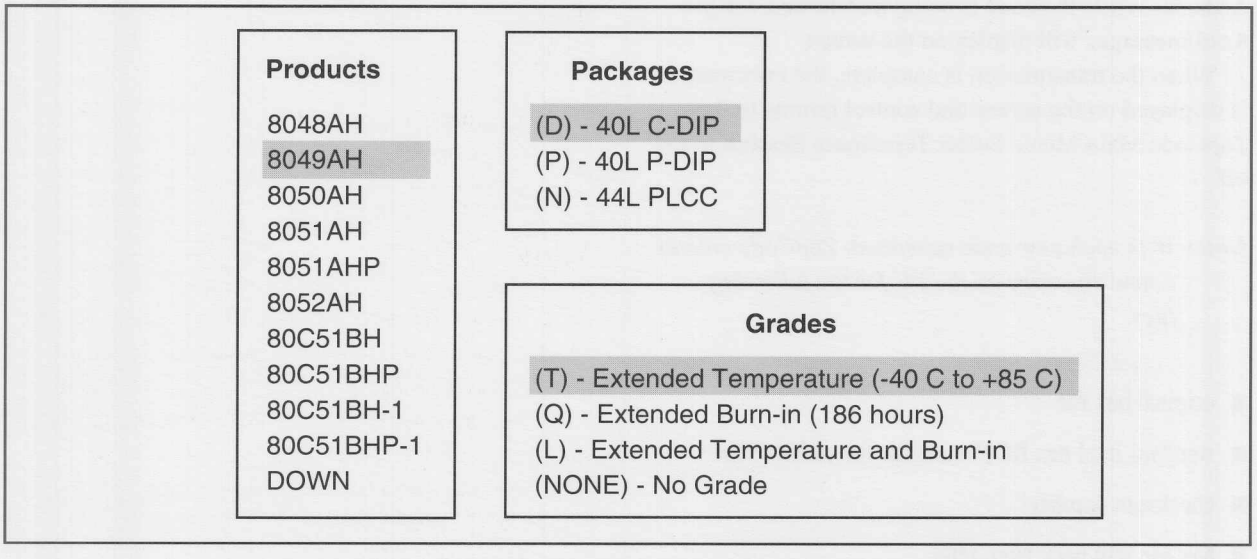

Figure 8. Products, Packages, and Grades Menu

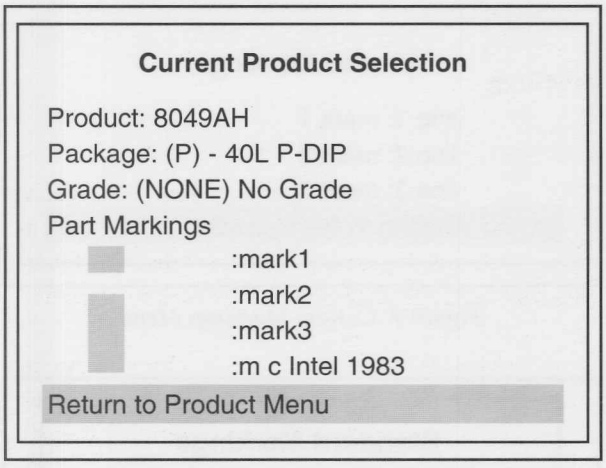

Figure 11. Example of Current Product Selection Screen

Continued from page 13

Note: *The Intel "i" and copyright information are legally required to appear on the parts.*

The **View Current Product Selection** option allows online viewing of the selected marking (Figure 11).

Select **Data Transmissions** from the Main Menu to begin transmitting the code to Intel (Figure 12).

The optional **[Comments]** selection allows entry of any special instructions for communication to Intel factory personnel.

The optional **[Application]** selection provides for a brief description of the application for this code.

The **Initiate Transmission** selection will start transmitting the code through the modem. The code is sent to our ROM Processing Area in Chandler, Arizona. While the code is being transmitted, the protocol messages will display on the screen.

When the transmission is complete, the information is displayed on the screen and control returns to the ZapCode Main Menu. Select **Terminate Session** to log off.

Note: *With each new code submitted, ZapCode creates a new directory on the PC for the following files:*

- original hex file
- standard Intel hex ROM code file for verification
- checksum number
- hex file sent back from Intel

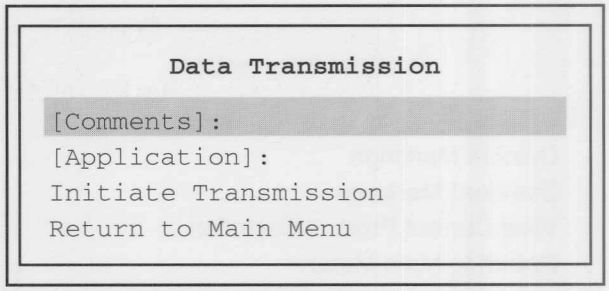

Figure 12. Data Transmission Menu

- customer letter, which contains the product number, ROM number, and part markings.

To compare the ROM code file that ZapCode sends back to you with the Intel hex-translated file you sent to Intel, type the following command after terminating the ZapCode session:

COMP old_filename new_filename

(example: COMP 1317.HEX CUST1.HEX)

To complete the process, print the customer letter, sign it, and FAX it (Attn: ROM Processing) to the number indicated on the letter.

ZapCode will let you know when your PC (ZAPPREP) files need to be updated. When this message appears, simply call ROM Processing for the updated version.

Remember, once ZapCode is installed, this entire process takes only minutes, and it saves our customers and Intel time and money!

MCS®-96 Asynchronous Serial Port

Larry C. Ferra
Technical Marketing Engineer
Intel Corporation
Article ID #0506

The routines discussed in this article are available from the BBS (File SERIALKC.C).

Asynchronous serial communication is a common peripheral function of microcontrollers. The 80C196 can generate a separate interrupt for a reception or a transmission. The iC-96 compiler is shipped with a library that polls the serial port to transmit and receive. This polling can waste considerable CPU bandwith at slow baud rates. Figure 1 is a sample interrupt-driven "putchar" and "getchar" that can be used in place of the library "putchar" and "getchar."

The size of the receive and transmit buffers can be set with the symbol definitions at the beginning of the program. The desired baud rate and operating frequency can also be set.

Continued on page 16

```
#pragma model(KD)
#include "80C196.h"
#pragma interrupt(receive=25,transmit=24)
#define TRANSMIT_BUF_SIZE 20
#define RECEIVE_BUF_SIZE 20
#define FREQUENCY 20000000L              /* 20 MHz  */
#define BAUD_RATE_VALUE 9600
#define BAUD_REG ((unsigned int)(FREQUENCY/((long)BAUD_RATE_VALUE*16)-1)+0x8000)
#define RI_BIT  0x40
#define TI_BIT  0x20
#ifdef EVAL_BOARD

/*   Reserve the 9 bytes required by eval board   */
char reserve[9];
#pragma locate(reserve=0x30)
#else
const unsigned int ccr = {0x20FF};   /*  Initialize the chip configuration bytes. */
#pragma locate(ccr=0x2018)
#endif

unsigned char status_temp;           /*   image of sp_stat  to preserve the RI */
                                     /* and TI bits on a read.  */

/*   receive and transmit buffers and their indexes    */
unsigned char trans_buff[TRANSMIT_BUF_SIZE];
unsigned char receive_buff[RECEIVE_BUF_SIZE];
char begin_trans_buff,end_trans_buff;
char end_rec_buff,begin_rec_buff;

/*  declares and locates the special function registers  */
void transmit(void)                  /*  serial interrupt routine  */
{
wsr = 0;
status_temp |= sp_stat;              /*  image sp_stat into status_temp  */

/*   transmit a character if there is a character in the buffer  */
if(begin_trans_buff!=end_trans_buff)
   {
   sbuf=trans_buff[begin_trans_buff];   /*  transmit character  */

/*     The next statement makes the buffer circular by starting over when */
/*                                          the index reaches the end of the
  buffer.   */
   if(++begin_trans_buff>TRANSMIT_BUF_SIZE - 1)begin_trans_buff=0;
   status_temp &= (~TI_BIT);         /*  clear TI bit in status_temp.   */
   }
}
void receive(void)                   /*  serial interrupt routine  */
{
wsr = 0;
status_temp |= sp_stat;              /*  image sp_stat into status_temp  */

/*   If the input buffer is full, the last character is ignored and the BEL charac-
       ter is output to the terminal.  */
if(end_rec_buff+1==begin_rec_buff || (end_rec_buff==RECEIVE_BUF_SIZE-1 &&
       !begin_rec_buff))
   {
   ;  /*  input overrun code  */
   }
else
   {

/* The next statement makes the buffer circular by starting over when  */
  /*                                 the index reaches the end of the buffer.  */
   if(++end_rec_buff > RECEIVE_BUF_SIZE - 1) end_rec_buff=0;
   receive_buff[end_rec_buff]=sbuf; /*    place character in buffer  */
```

Figure 1. Interrupt-driven "putchar" and "getchar" Example

```
    }
status_temp &= (~RI_BIT);      /*  clear RI bit in status_temp.  */
}
int putchar(int c)
{

/*                                              remain in loop while the buffer is
  full.  This is done by checking the end of buffer index to make sure  */
/*                                              it does not overrun the beginning of
  buffer index. The while instruction checks the case in which the    */
/*                                              end index is one less than the begin-
  ning index and at the end of the buffer when the beginning index */
/* may be equal to 0 and the end buffer index may be at the buffer end.    */
while((end_trans_buff+1==begin_trans_buff)||
    (end_trans_buff==TRANSMIT_BUF_SIZE -1 && !begin_trans_buff));
trans_buff[end_trans_buff]=c;          /*  put character in buffer  */
if(++end_trans_buff>TRANSMIT_BUF_SIZE - 1)      /*  make buffer appear circular */
    end_trans_buff=0;
if(status_temp & TI_BIT) ipend1 |= 0x01;/*  If transmit buffer was empty, then cause
       an interrupt to start transmitting.  */
}
unsigned char getchar()
{
while(begin_rec_buff==end_rec_buff); /*  remain in loop while there is no character
       available. */
if(++begin_rec_buff>RECEIVE_BUF_SIZE - 1)       /*  make buffer appear circular  */
    begin_rec_buff=0;
return(receive_buff[begin_rec_buff]);    /*  return the character in buffer.  */
}
void init_serial_port()
{
unsigned char wsr_image = wsr;
wsr=0;
baud_rate = ((unsigned char) BAUD_REG);
baud_rate = ((unsigned char) (BAUD_REG >> 8));
sp_con = 0x09;                          /* mode 1, no parity, receive enabled, no 9th
       bit */
status_temp = sp_stat | TI_BIT;

end_rec_buff=0;                         /* initialize buffer pointers        */
begin_rec_buff=0;
end_trans_buff=0;
begin_trans_buff=0;
imask1=0x03;                            /* enable the serial port interrupts  */
enable();                               /* global enable of interrupts       */
wsr = wsr_image;
}
main()
{
char c;
init_serial_port();
while((c=getchar()) != 0x1b)            /*  stay in loop until escape key is pressed
       */
    printf("key pressed = %02X\n\r",c);
}
```

Figure 1. Interrupt-driven "putchar" and "getchar" Example (Continued)

New Embedded Control FaxBACK® Service Documents

As of May 4, 1993

Title	Number
80C186 Product Information Line Card	2000
80C51SL-BG Errata, Version 2.6	2008
EPROM Socket Adapters (for devices in non-DIP packages)	2202
Programmer Vendor Addresses & Phone Numbers (all programmable devices)	2203
Flash Memory Socket Adaptors	2574
28F256A Programming Support	2575
28F512 Programming Support	2576
28F010 Programming Support	2577
28F020 Programming Support	2578
28F001BH Programming Support	2579
28F002BX/200BX Programming Support	2580
28F004BX/400BX Programming Support	2581
28F008SA Programming Support	2582
8-Bit Microcontroller Programming Support 8748, 8749	2585
8-Bit Microcontroller Programming Support 8751BH, 8752BH	2586
8-Bit Microcontroller Programming Support 87C51, 87C51FX, 87C54, 87C58	2587
16-Bit Microcontroller Programming Support MCS®-96 Family	2588
MCS®-96 Family Product Information Line Card	2591
MCS®-51 Architecture Development Tools Line Card	2622
MCS®-96 Architecture Development Tools Line Card	2623
80C186 Development Tools Line Card	2624
8XC194, 8XC198, and 8XC196KB Fact Sheet	2626
Master Embedded BBS File Listing	2627
Intel MCS®-51 CTO Product Status Update, Oct. '92	2628
80C31BH/80C51BH ESD Testing	2629
80C51SL-BG Product Preview Data Sheet Errata	2630
8X97BH, 8X97JF Reserved Memory Location 2019H	2631

New/Updated Articles as of April 2, 1992

Title	Number
Interfacing Floppy Disk Controller to 186EX Family	2168
8XC196NT Errata	2178
80C186 Third-Party Real-Time Kernels	2179
Flash Memory Card Programming Support	2200
Memory Card Drives, Card Reader/Writers	2201
186 Family User's Manual Errata	2603
(AB-53) Low Output Skew/Clock Driver Applications of the 85C224 Family	2604
iPLD22V10-7 Preliminary Data Sheet	2605
3.3 volt iPLDLV22V10-15 Preliminary Data Sheet	2606
(AB-54) Using the iFX780 FPGA Features with PLDshell Plus R3.0	2607
(AB-55) How to Use Registered SRAM Macrocells on the iFX780 FPGA	2608
(AP-390) JTAG 1149.1 Specs for iFX780 (Part 1)	2609
(AP-390) JTAG 1149.1 Specs for iFX780 (Part 2)	2610
80960CA (C-3 Step) Technical Bulletin #11.0	2611
80960CA (B-0 Step) Technical Bulletin #4.0	2612
80960CX DMA AC Timing Scenarios	2613
80960SA/SB Technical Bulletin	2614
80960SA/SB User's Manual Errata	2615
80960KA/KB Programmer's Manual Errata	2616
80960KB Hardware Designer's Reference Manual Errata	2617
82961KA (A-0, A-1 Step) Errata	2618
82961KD (A-0 Step) Errata	2619
80960 Eval Board Repair/Return Policy	2620
80960 Benchmarking from Your Office	2621

Vertical Windows on the 80C196KC/KD

Marty Goodman
Applications Engineer
Intel Corporation
Article ID #0507

Vertical Windows (VWindows) are used to map sections of the Register RAM into the upper section of the lower Register File. The 80C196KC has 512 bytes (00H–1FFH) and the 80C196KD has 1024 bytes (00H–3FFH) of Register RAM that may be remapped.

An important difference between Horizontal and Vertical Windows is that VWindows reside directly in the 80C186KC/KD addressing space. In other words, while HWindows just swap 24-byte chunks of Special Function Registers into and out of the lower Register File, VWindows actually remap locations within the general-purpose register RAM address space. With HWindows, you use only register-direct addressing (and get the Special Function Registers located within the HWindow), but with VWindows you can use either a register-direct address (and get the location within the VWindow) or a 16-bit indirect or indexed address (and get the actual general-purpose register RAM location in memory).

VWindows may be one of three sizes: 32, 64, or 128 bytes. The Window Select Register (WSR) determines both the size of the VWindow and which window of that size is to be remapped. Figure

Continued on page 20

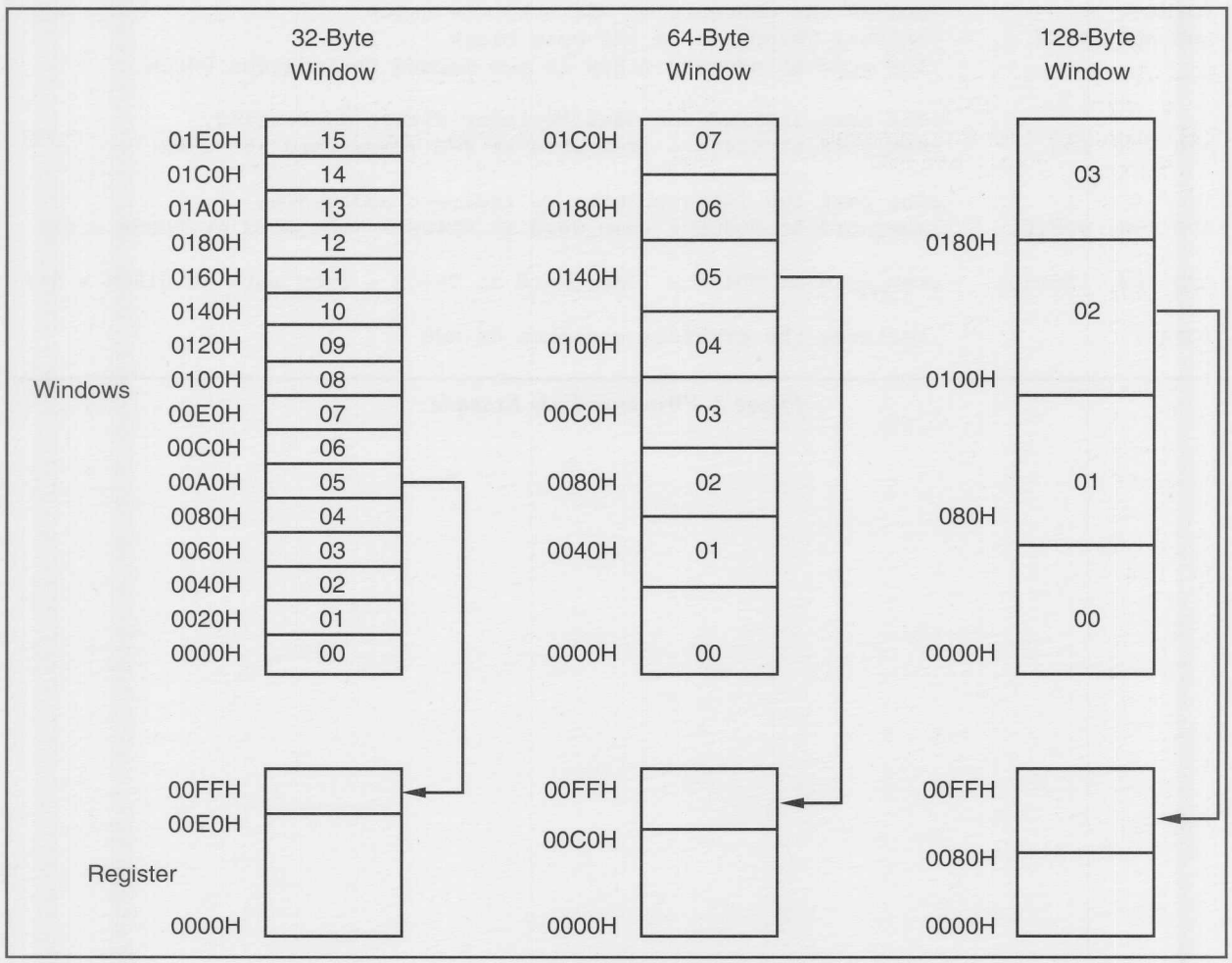

Figure 1. VWindows

Continued from page 19

1 shows all the available VWindows on the 80C196KC. The 80C196KD has twice as many windows, extending from 00H to 3FFH.

When a 32-byte window is selected, it is placed at E0H and extends to FFH. When a 64-byte window is selected, it is mapped from C0H to FFH; and when a 128-byte window is selected, it is mapped from 80H to FFH. The Register RAM that is "covered" by the window is not accessible by register-direct addressing when VWindows are active. However, that "covered" memory in the lower Register File is still accessible using 16-bit indirect or indexed addressing.

As an example, let's map the 128-byte block from 180H–1FFH into the upper part of the Register File from 80H–FFH. Now any access to locations 80H–FFH using a register-direct reference will actually access the memory at 0180H–01FFH. However, accesses to locations 0080H–00FFH using 16-bit addressing will access the registers as if the VWindow were not active.

The sample code in Figure 2 shows the VWindow being switched, as well as the correct value to load into the WSR. It also illustrates the difference between register-direct and indexed addressing when using VWindows.

VWindows provide for fast context switching of register sets. For example, an Interrupt Service Routine could have its own set of local registers in a VWindow and pass results to a main routine through global registers in the Register File.

The Window Select Register (WSR) description in Appendix C of the user's manual (both printed and electronic versions) includes tables that explicitly list the remapped memory sections corresponding to specific WSR values.

```
PUSHA                ;pushes the contents of WSR onto the stack
LDB WSR, #13H        ;selects VWindow 3, a 128-byte block
                     ;The word at address 0180H is now mapped to location 0080H.

                     ;The next instruction uses register-direct addressing.
ADD 40H, 80H         ;mem_word at 0040H = (mem_word at 40H + mem_word at 0180H)

                     ;The next two instructions use indirect addressing.
ADD 40H, 80H[0]      ;mem_word at 0040H = (mem_word at 0040H + (mem_word at 0080H + 0))

ADD 40H, 180H[0]     ;mem_word at 0040H =  (mem_word at 0040H + (mem_word at 0180H + 0))

POPA                 ;restores the previous contents of WSR
```

Figure 2. VWindows Code Example

BHE# Functionality During Bus Cycles

Richard N. Evans
Applications Engineer
Intel Corporation
Article ID #0508

The MCS®-96 devices have a time-multiplexed address/data bus. The data bus can be configured to be either 8 bits wide or 16 bits wide. This article explains how to use the byte high enable pin (BHE#).

BHE# and a 16-bit Bus Width

When the bus width is configured to be 16 bits, words can be written 16 bits at a time. But what happens when just a high byte or low byte is written with a store byte (STB) instruction? The memory controller puts out the given byte on the upper data lines (AD8-AD15) and lower data lines (AD0-AD7). So how are the upper and lower bytes discerned by the memory system? The MCS-96 devices provide two ways of writing individual bytes (i.e., STB) with a 16-bit bus. One way the MCS-96 device provides is using the byte high enable (BHE#) signal. By combining A0, WR#, and BHE# with logic, a WRL# and a WRH# signal can be created. These are the logic equations:

WRL# = (A0 + WR#)

WRH# = (BHE# + WR#)

The most convenient way of generating a WRL# and WRH# signal is by using Write Strobe mode. This mode can be selected with a bit in the chip configuration register as an alternative to using the BHE# signal. The WRH# signal shares a pin with the BHE# signal, and the WRL# signal shares a pin with the WR# signal. CCB.2 is the select bit for the multiplexed pins. Setting this bit (1) selects BHE# and WR#. Clearing the bit (0) selects the Write Strobe mode, in which WRL# and WRH# are generated instead of BHE# and WR#.

The BHE# signal is valid during both read and write cycles. When a byte read (i.e., LDB) is initiated with a 16-bit bus width, the MCS-96 reads a word and discards the unwanted byte.

Odd and Even Memory Banks

Often two 8-bit wide memory devices are combined to create a 16-bit data bus. These two banks of memory can be split into even and odd addresses. The BHE# signal and address line 0 (A0) can be used as chip selects, with BHE# connected to the odd bank and A0 connected to the even bank. It is important to remember that the address lines and the bus control signals are not driven during internal execution, so these signals should be latched via ALE. For a summary of how BHE# responds in different configurations, see Table 1.

This article applies to all MCS-96 devices. For more information about memory interfacing and BHE#, see the *8XC196KC/KD User's Manual* (order # 272238) or the *8XC196Kx Family User's Manual* (order #272258).

Table 1. BHE# State

Buswidth	Access	BHE# State	Read	Write
8	Word	Asserted	Bytes on low data lines — first low, then high	Bytes on low data lines — first low, then high
8	High Byte	Asserted	Byte on low data lines	Byte on low data lines
8	Low Byte	Deasserted	Byte on low data lines	Byte on low data lines
16	Word	Asserted	Word on AD0–AD15	Word on AD0–AD15
16	High Byte	Asserted	Word on data bus, low byte discarded	Same byte on high and low data lines
16	Low Byte	Deasserted	Word on data bus, high byte discarded	Same byte on high and low data lines

Q's and A's on the 186EX Evaluation Boards

Steven Leedom
Applications/Tools Technician
Intel Corporation
Article ID #0509

This list answers a few of the most frequently asked questions about the 186EX evaluation boards.

Q: I would like to use 80C188XX instead of the 80C186XX in my Eval board. What do I do?

A: Replace the 80C186XX with the 80C188XX chip on the Eval board. Set switch 1 of S2 to the ON position to let the Eval boards work for 8-bit bus width.

Q: How do I interface an 80C187 to my Eval board?

A: Place the 80C187 coprocessor in the socket provided and an oscillator in its socket. For the EA/XL, set switch 4 of S2 to the ON position. The EB board will automatically detect the 80C187. For the EC, set switch 2 of S2 to the ON position.

Q: Where can I find the iRISM and iECM software? Can I use them in my design?

A: The iRISM and iECM are supplied with the Eval board kit. Both are on the BBS under Embedded Controllers, 186 files (these files are zipped text and executable). Yes, you can use them in your design as needed. The iRISM source code is supplied with the kit and is also available on the BBS. The iECM source code is not available.

Q: How can I get the Eval board schematics?

A: All the schematics (in OrCAD* format) are on the BBS.

Q: I have a problem with my Eval board and it does not seem to function at all. Is there a warranty?

A: Yes. All Eval boards are under a one-year warranty, only if customers have returned their warranty card. Any board that has been modified by the customer will no longer be under warranty.

Q: If my Eval board is under warranty and does not work, who do I send it to for repair or replacement?

A: Before any board is returned, the full problem needs to be documented and understood. Call Customer Support for more information.

Q: I lost the software diskette that came with the Eval board. How do I get another copy?

A: The software is on the BBS under Embedded Controllers, 186 files.

Q: How can I get a copy of the Paradigm software?

A: If you order the DK80C186XX kit, you get the Paradigm software with the Eval board. To get just the Paradigm software, call Paradigm directly (800-537-5043). For more information, request document #2086 from the FaxBACK® service.

*OrCAD is a registered trademark of OrCAD Inc.

*Ap*BUILDER Version 1.2

The *Ap*BUILDER software version 1.2 adds full support for eight new devices: the 80C186XL, 80C196NT/NQ, and 80C196KR/JR/KQ/JQ/KT. This new version generates peripheral code as function calls and uses macros for register access. Among its many enhancements, the new version has

- an online guided tutorial, called Quick Tutor
- an improved, graphical help system
- streamlined Facts section, with more information
- better display resolution and performance
- user-selectable colors for some parts of the screen

You can download *Ap*BUILDER v1.2 from the BBS or you can order it. In the U.S. and Canada, call Intel Literature at 800-468-8118 and order #272216-002. In Europe and other international locations, please contact your local Intel sales office or distributor.

Q's & A's

Article ID # 0510

80C18X Family Q's & A's

Q: When will the 80C18xXL C-step be available and how will it differ from the B1-step?

A: C-step samples are available now, and production will be available in May '93. The C-step will fix the INTx/INTAx errata, which is an internal problem with the interrupt controller. This problem can prevent an acknowledge cycle on the INTA1 pin after an interrupt on INT1 or prevent an acknowledge cycle on the INTA0 pin after an interrupt on INT0. It occurs when one interrupt is configured in cascade mode and a higher priority interrupt occurs after the decision is made to service the lower priority interrupt but before the expected acknowledge cycle on the corresponding pin.

For details, request document #2025, "80C18xXL/EA/EB INTx/INTAx# Errata," from the FaxBACK® service.

Q: What is included in the DK80C186Ex design kits?

A: The design kits include a 186 evaluation board, Paradigm software development kit (debug/RT, Turbo Debugger), Paradigm Locate, starting code, run-time library, floating-point support, and working examples. For more information, request document #2086, "DK80C186EA/XL, EB, & EC Design Kit Overview," from the FaxBACK service.

MCS®-51 Family Q's & A's

Q: If you stop the clock during RESET, will RESET continue when the clock is reapplied?

A: RESET must be held high for a minimum of 24 oscillator periods after the clock returns for a valid RESET to occur.

Q: Can you poll the 16-bit timer overflow flag and clear it without using an interrupt?

A: Yes, disable the timer interrupt and then poll the appropriate timer overflow flag in TCON. The bit can then be cleared with a clear bit command.

Q: What should be done with unused I/O ports?

A: Set unused ports as outputs and drive them high.

Q: Does Port 0 require external pull-up resistors?

A: External pull-ups are required only if the port is being used to drive an output.

MCS®-96 Family Q's & A's

Q: What evaluation board is used to debug code for the 80C198?

A: There is no longer an evaluation board for the 80C198. We suggest that you use the 80C196KB Eval Board with the KB chip to debug code for a 198 design because of their similarities.

Q: What is the recommended development path for the 83C152 without an EPROM version available?

A: Develop using the CPU part of the 83C152 and an external EPROM. Once the design is debugged, convert to the factory-masked ROM part that is available.

Q: Is the 80C196KC static, or will data be lost when the clock is stopped?

A: The part is fully static; however, it is only tested between 8 and 16MHz. It will typically operate below 1Hz.

Q: Can the synchronous serial port of the 8XC196Kx family devices be used in an asynchronous mode?

A: No, you cannot use the SSIO ports to function as asynchronous ports. If you need two ports for a low-speed port with a fixed baud rate, use the asynchronous port and generate the second in one of two ways:

1. Use two EPA modules to emulate the port. They are very similar to the PCA found on the 51FX core products. Application Brief AB-41 in the *Embedded Applications* handbook describes using the PCA in this manner.
2. Implement the port with regular port pins. There are no design applications for this implementation.

Flash Memory Q's & A's

Q: What is the part number for the ExCA™ Hardware Developer's Kit?

A: The EXCAHWEBD allows you to evaluate the benefits of Intel's Series 1 and Series 2 Flash Memory Cards. The ExCA™ Hardware Developer's Kit supersedes Intel's PCCARDKIT and Flash Memory Systems Developers Kit (iMSDK001FLKA). The Product Brief order number is #297293.

Errata and Change Identifiers

Article ID #0511

Change identifiers have been used since 1990 to distinguish revisions, or steppings, of embedded control devices. Older devices have no change identifiers. On most devices, the change identifier is the last character in the FPO number, which is typically a nine-character code on the second line on the top of the device. An example FPO number is "L1234567D," in which "D" is the change identifier. On some devices, such as the 8XC51SL-BG, the change identifier is a separate line item and uses several characters. For example, change identifier "SW011" identifies the B-3 stepping of the 8XC51SL-BG.

This article lists change identifiers, errata, and design considerations for recent steppings of embedded control products. For many of these devices, complete errata listings or explanations are available from the FaxBACK® service. The "Ref" column in each table lists FaxBACK service document numbers for errata lists and explanations, and "For More Information" at the end of this article has a complete list of related document numbers and titles.

MCS®-51 Family Errata and Design Considerations

This list covers the most recent steppings of the MCS®-51 family devices. A complete list of the errata for all steppings is available on the FaxBACK service (document #2632).

Table 1. MCS®-51 Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8051AH/8031AH	C	A	1. External interrupt 0 errata	2154 2161
	C-3	B	No known errata	
80C51BH/80C31BH	C	none	1. Reset lockup problem 2. High IPD if C does not equal B.7 before going into powerdown 3. ROM verify mode fails 4. Steam passivation problem on plastic parts	
	C-1	none	1. High IPD if C does not equal B.7 before going into powerdown 2. ROM verify mode fails	
	D	D or 2	No known errata	
87C51	D	A	No known errata	2106
80C52/80C32	C	none	1. RST/ONCE mode problem	
	A	A	No known errata	
83C51FA/80C51FA	C	none	1. PCA errata	2528
87C51FA	C	none	1. RST/ONCE mode problem 2. PCA errata	2528
	D	A	No known errata	2107
8XC51FB	A	none	1. PCA errata	2528
	B	A	No known errata	2111

Table 1. MCS®-51 Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC51FC	A	none	1. Port 1, 2, 3 problem — asychronous port reset not supported 2. Failed ESD qual testing	2028
	B	none	No known errata	
8XC51GB	B	none	1. Reset polarity changed to active low 2. Port 1 reset value changed to all zeros	2032 2032
	B-2	none	No known errata	
8XC152JX	B	none	1. AE/RDN race condition 2. Receive FIFO is not cleared when receiver is enabled 3. DMA errata 4. SDLC flag recognition errata	 2030 2118 2035
	C	none	No known errata	
8XC51SL-BG	B-3	SW011	1. GATEA20, RCL hardware speedup processing 2. Powerdown current stabilization 3. Port 2 address mux 4. KSI powerdown wakeup interrupt 5. Reset errata	2008 2114
	B-4	SW062	1. GATEA20, RCL hardware speedup processing 2. Powerdown current stabilization 3. Port 2 address mux 4. KSI powerdown wakeup interrupt	2008

MCS®-96 Family Errata and Design Considerations

This list covers the most recent steppings of the MCS®-96 family devices. Complete lists of the errata for all steppings are available on the FaxBACK® service for the 8X9XBH (#2134), the 8XC196KB (#2548), the 8XC196KC (#2136), the 8XC196KR (#2527), and the 8XC196NT (#2178).

Table 2. MCS®-96 Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8X9XBH	D	D	1. Indexed 3-operand multiply 2. HSI FIFO 3. Reserved location 2019H 4. RESET and the QBD pins 5. Software RESET timing 6. Using T2CLK for Timer2	2134
	E	E	1. Indexed 3-operand multiply 2. HSI resolution 3. Reserved location 2019H 4. Reserved location 201CH	 2631 2140
8XC196KB 8XC196KB10/KB12	B	B	1. Divide during HOLD or READY	2548

Table 2. MCS®-96 Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC196KB 8XC196KB10/KB12 (Continued)			2. HSI 8/9 state 3. SIO framing error 4. SIO RI flag 5. DJNZW instruction 6. CMPL with R0 7. ALE glitch	 2156 2568
8XC196KB/KB16	B	D	1. Divide during HOLD or READY 2. HSI 8/9 state 3. CMPL with R0	2122 2156
	C	E	1. HSI 8/9 state 2. CMPL with R0	 2156
8XC196KC			The number following each entry in this list isa cross-reference to the applicable section of FaxBACK® service document #2136.	2136
	all		Design Considerations 1. Indirect shift count value 116 2. Write cycle during Reset 147	
	A	A or none	**Errata** 1. A/D convert error 101 2. BMOVI instruction 105 3. Divide error during hold 109 4. HSO IOC1 bits interchanged 115 5. Port 0 latched on wrong phase 126 6. PTS Req during INT latency 131 7. NMI during PTS latency 132 8. SIO Mode 0 140 9. TIJMP INDEX_MASK value 143 10. Serial Port Framing Error 150 11. QBD glitch during powerup 163 12. A/D linearity too large 173 13. Analog input latch-up 174 14. INST weak during CCB fetch 175 15. 2 CCB fetches 176 16. 3 wait states during CCB 177 17. Pullups too weak during Reset 178 18. Buswidth always 16 bits 179 19. Vcc glitch resets device 180 20. Incomplete reset at > 12 MHz 181 21. Oscillator startup 215 22. Reset hysteresis 216	
	B-1	B	1. Divide error during hold 109 2. NMI during PTS skips address 123 3. QBD glitch during powerup 163 4. ONCE mode entry 214 5. Oscillator startup 215 6. Reset hysteresis 216	
	B-3	D or E	1. Divide error during hold 109 2. NMI during PTS skips address 123 3. QBD glitch during powerup 163 4. Reset hysteresis 216	
8XC196KD	A-1	B	No known errata	

Table 2. MCS®-96 Family Errata and Design Considerations (Continued)

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
8XC196KR/JR/KQ/JQ	A, C	A, C	Design Considerations 1. P6_REG not updated immediately 2. Write cycle during reset 3. EPA timer reset/write conflict 4. Valid time matches 5. CLKOUT 6. Indirect shift operation 7. Internal RAM powerdown leakage 8. A/D latchup 9. INST operation 10. KQ/JQ memory map	2527
	A	A	**Errata** 1. Oscillator noise sensitivity 2. Slave programming mode 3. A/D abort 4. PTS with other interrupts 5. PTS/NMI conflict 6. Data output register cleared when mode register is written 7. Divide error during Hold/Ready 8. SIO Mode 0 9. Remap mode on EPA3 10. Serial port framing error 11. EPAIPV value multiplied by 2 12. EPA_MASK1/EPA_PEND1 must be written as words 13. Interruptable block move (BMOVI)	
	A, C	A, C	1. Ioh2 = – 6 µA	
8XC196NT	C	C	1. eld(b), est(b) in auto-increment mode over 64K boundaries 2. Extended base indexed eld(b), est(b) accessing memory when executing from external memory 3. In bus controller modes 1 and 2, in 8-bit bus mode, the upper address lines need to be latched	2178

Table 3. 8XC186/8XC188 Family Errata and Design Considerations

Device	Step	Change Identifier	Errata and Design Considerations	Ref.
80C186A, B	none		1. Non-contiguous Interrupt Acknowledge cycles 2. ERROR# processing during FWAIT instructions 3. Input high voltage requirement on SRDY and ARDY pins 4. Interrut Status Register (DHLT and Timer Interrupts) 5. Bus preemption errata (HOLD/HLDA protocol) 6. 80C188 RFSH# pin output timing	2096
80C186XL	A	none	Never put into production	
	B	A	1. INTx/INTAx in Cascade Mode	2025
	C	B	No known errata	
80C186EA/80L186EA	A	A	1. Low hysteresis on RESIN# pin 2. TEST/BUSY#, RD#/QSMD#, LCS#, and UCS# input low voltage 3. INTx/INTAx in Cascade Mode	2025
	B	B	1. INTx/INTAx in Cascade Mode	2025
80C186EB/80L186EB	A	A or none	1. Entry into ONCE mode 2. Low hysteresis on RESIN# pin 3. SINT1 input not latched internally 4. Ready input during INTA# bus cycle 5. CLKOUT transitions on the rising of CLKIN instead of the falling edge 6. I/O ports 1 and 2 initialize to Port instead of Peripheral function (documentation error) 7. INTx/INTAx in Cascade mode	2025
	B	B	1. INTx/INTAx in Cascade Mode	2025
80C186EC	A	A	1. Early exit from Reset (with high Vcc, or low temperature, or both) 2. Powersave Mode initialization at Reset	
	B	B	No known errata	

For More Information

The following FaxBACK® service documents contain errata lists and explanations.

MCS®-51 Family

Title	Number
8051/31AH Shrink C-Step External Interrupt 0 Errata	2161
8051/31AH Shrink Conversion	2154
80C31/51BH D-Step Marking	2015
80C51SL-BG Errata	2130
80C51SL-BG Errata, Version 2.6	2008
80C51SL-BG Product Preview Data Sheet Errata	2630
80C51SL-BG Reset Errata	2144
87C51 D Step	2106
87C51FA D Step	2107
87C51FB/83C51FB B Step	2111
87C51FC Data Sheet Errata I	2024
87C51FC Data Sheet Errata II	2020
87C51FC Port Anomaly	2028
87C51GB A-1 & B Errata & Design Considerations	2032
87C54/80C54 Data Sheet Errata I	2022
87C54/80C54 Data Sheet Errata II	2021
8XC152 DMA Bug	2118
8XC152 Global Serial Channel Bug	2030
8XC152 SDLC Flag Recognition Bug	2035
8XC51FA/FB/FC Hardware Description Errata	2017
8XC51FX PCA/Timer2 Errata	2528

MCS®-96 Family

8X97BH, 8X97JF Reserved Memory Location 2019H	2631
8X9X Reading 201CH Bug	2140
8X9X SIO Status Register Errata	2138
8X9XBH, 8X9XJF, and 8X98 Bug History	2119
8X9XBH, 8X9XJF, and 8X98 Conversions	2133
8X9XBH, 8X9XJF, and Bug List	2134
80C196 Oscillator Noise Sensitivity	2113
8XC196 Hold/Ready DIV, DIVB Problem	2122
8XC196KB and 8XC198 Bug List	2135
8XC196KB History and Errata	2548
8XC196KB ALE Glitch	2568
8XC196KB/198/194 CMPL Instruction Using R0	2156
8XC196KC 1991 Handbook Errata	2131
8XC196KC Bug List	2136
8XC196KC HSI PTS Handbook Errata	2141

MCS®-51 Family (Continued)

Title	Number
8XC196KC/KD 1992 User's Manual Errata	2570
8XC196KR Errata and Design Considerations	2125
8XC196KR/JR/KQ/JQ Errata	2527
8XC196NT Errata	2178
8XC198 and 8XC196KB/KB16 Data Sheet Errata	2569
Itl Specification Change on 8XC198, 8XC196KB/8XC196KB16	2567

80C186/80C188 Family

186 Design Spurious Writes	2553
186 Family Eval Board Errata	2592
186 Family User's Manual Errata	2603
80C186/188 Bus Preemption Problem	2096
80C186/80C188 Interrupt Controller Error	2025
80C186/C188 B-Step Technical Bulletin	2100
80C186/C188EA A-Step Technical Bulletin	2102
80C186/C188EB A-Step Technical Bulletin	2103
80C186/C188EB B-Step Technical Bulletin	2104
80C186/C188EC A-Step Technical Bulletin	2105
80C186/C188XL B-Step Technical Bulletin	2101
80C18x XL/EA/EB INTx/INTAx# Errata	2025
EV80C186EB Eval Board Manual Errata	2158

Sales Offices and Distributor Addresses

Intel Corporation
Literature Department
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
United States

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Japan

Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-
Yvelines Cedex
France

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ
United Kingdom

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Germany

Intel Semiconductor Ltd.
32/F Two Pacific Place
88 Queensway, Central
Hong Kong

Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Canada

U.S. INTEL SALES OFFICES — ***Call (800) 628-8686 to contact any of the following U.S. sales offices:***

Alabama, Huntsville; **Arizona,** Phoenix; **California,** Roseville, San Diego, San Jose, Santa Ana, Sherman Oaks; **Colorado,** Denver; **Connecticut,** Danbury; **Florida,** Deerfield Beach, Maitland; **Georgia,** Norcross; **Illinois,** Schaumburg; **Indiana,** Indianapolis; **Maryland,** Annapolis Junction; **Massachusetts,** Westford; **Michigan,** West Bloomfield; **Minnesota,** Bloomington; **New Jersey,** Red Bank; **New York,** Fairport, Fishkill, Islandia; **Ohio,** Beachwood, Dayton; **Oklahoma,** Oklahoma City; **Oregon,** Beaverton; **Pennsylvania,** Blue Bell; **South Carolina,** Columbia, Greenville; **Texas,** Austin, Dallas, Houston; **Utah,** Murray; **Washington,** Bellevue, Spokane; **Wisconsin,** Brookfield

CANADIAN SALES OFFICES — ***Call (800) 628-8686 to contact any of the following Canadian sales offices:***

British Columbia, Intel Semiconductor of Canada, Ltd., Vancouver
Ontario, Intel Semiconductor of Canada, Ltd., Ottawa; Intel Semiconductor of Canada, Ltd., Rexdale
Quebec, Intel Semiconductor of Canada, Ltd., Pt. Claire

U.S. DISTRIBUTORS:

Alliance Electronics, Inc. / Almac/Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / MTI Systems / MTI Systems Sales / North Atlantic Industries Systems Division / Pioneer-Standard / Pioneer Technologies Group / Wyle Laboratories

CANADIAN DISTRIBUTORS:

Almac-Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / Zentronics

EUROPEAN INTEL SALES OFFICES:

Finland, Helsinki, (358) 0 544 644
France, St. Quentin-en-Yvelines Cedex, (33) (1) 30 57 70 00
Germany, Feldkirchen bei Muenchen, (49) 089/90992-0
Israel, Tel-Aviv, (972) 03 498080
Italy, Assago, (39) (2) 575441
Netherlands, Rotterdam, (31) 10 407 11 11
Russia, Moscow, 007-095-4439785
Spain, Madrid, (34) (1) 308 2552
Sweden, Solna, (46) 8 705 5600
United Kingdom, Swindon, (44) (0793) 696000

EUROPEAN DISTRIBUTORS:

Austria, †*Bacher Electronics GmbH, Wien, (43) 1816020
Belgium, †*Inelco Distribution, Bruxelles, (32) 2 244 2811; *Diode Belgium, Zaventem, (32) 2 725 46 60
Denmark, *Avnet Nortec A/S, Herlev, (45) 4284 2000; †*ITT Multikomponent AS, Glostrup, (45) 4245 6645
Finland, †*OY Fintronic AB, Helsinki, (358) 0 682 791
France, *Arrow Electronique, Rungis Cedex, (33) (1) 4978 4978; †Metrologie, Asnieres Cedex, (33) (1) 4080 9000; *Tekelec, Sevres, (33) (1) 4623 2425
Germany, *Electronic 2000, Muenchen, (49) 89 42110-01; *Jermyn GmbH, Limburg, (49) 6431 5080; †Metrologie GmbH, Muenchen, (49) 89 724470; *Proelectron Vertriebs GmbH, Dreieich, (49) 6103 304343; †Rein Elektronik GmbH, Nettetal, (49) 2153 7330
Greece, †Ergodata, Kalithea, (30) 1 95 10 922; *Pouliadis Associates Corp., Athens, (30) 1 36 03 741
Ireland, †*Micro Marketing, Dublin, (353) (1) 298 9400
Israel, †*Eastronics Limited, Tel-Aviv, (972) 3 6458 777
Italy, *Intesi Div. Della Deutsche — Divisione ITT Industries GmbH, Assago (Milano), (39) 2 824701; *Lasi Elettronica, Milano, (39) 2 661431; †Omnilogic Telcom, Milano, (39) 2 48302640
Netherlands, †Datelcom, BD Maarsen, (31) 3465 95222; *Diode Components, NG Nieuwegein, (31) 3402 9 12 34; †*Koning en Hartman, AP Delft, (31) 15 609 906
Norway, *Avnet Nortec A/S, Hvalstad, (47) 284 6210; †Computer System Integration A/S, Skjetten, (47) 6 84 54 11
Portugal, *ATD Electronica LDA, Lisboa, (351) (1) 847 2200; †Metrologia Iberica Portugal, Lisboa, (351) (1) 847 2202

EUROPEAN DISTRIBUTORS (Contd.):

South Africa, †*EBE, Pretoria, (27) 12 803 7680-93
Spain, *ATD Electronica, Madrid, (34) (1) 661 6551; †Metrologia Iberica, Madrid, (34) (1) 661 1142
Sweden, †Avnet Computer AB, Farsta, (46) 8 705 18 00; *Avnet Nortec AB, Solna, (46) 8705 1800; *ITT Multikomponent AB, Solna, (46) 8 830020
Switzerland, †IMIC Microcomputer, Winkel-Ruti, (41) (1) 8620055; †*Industrade A.G., Wallisellen, (41) (1) 8328111
Turkey, *Empa Electronic, Istanbul, (90) (1) 599 3050
United Kingdom, *Arrow Electronics, Bedford, (44) 234 270272; *Avnet EMG Ltd., Hertsfordshire, (44) 462 488 500; *Bytech Components, Hants, (44) 256 707 107; †Bytech Systems, Berks, (44) 344 55 333; *Jermyn Electronics, Kent, (44) 732 743 743; †Metrologie VA, Bucks, (44) 494 526 271; *MMD/Rapid Ltd., Berks, (44) 734 750 697

INTERNATIONAL SALES OFFICES:

Australia, Intel Australia Pty. Ltd., Sydney, 61-2-975-3300; Intel Australia Pty. Ltd., Melbourne, 61-3-810-2141
Brazil, Intel Semiconductores do Brazil LTDA, Sao Paulo, 55-11-287-5899
China/Hong Kong, Intel PRC Corp., Beijing, (1) 500-4850; Intel Semiconductor Ltd., Hong Kong, (852) 844-4555
India, Intel Asia Electronics, Inc., Bangalore, 91-812-215065
Japan, Intel Japan K.K., Tsukuba HQ, 0298-47-8511; Intel Japan K.K., Tokyo HQ, 03-3201-3621; Intel Japan K.K., Tokyo, 03-3493-6081; Intel Japan K.K., Kanagawa, 045-474-7660; Intel Japan K.K., Osaka, 06-863-1091; Intel Japan K.K., Hachioji-shi, 0426-48-8770
Korea, Intel Korea, Ltd., Seoul, (2) 784-8186
Mexico, Intel Tecnologia de Mexico S.A. de C.V., Guadalajara, Jal., 523-640-1259
Singapore, Intel Singapore Technology, Ltd., (65) 250-7811
Taiwan, Intel Technology Far East Ltd., Taipei, 886-2-5144202

INTERNATIONAL DISTRIBUTORS:

Argentina, Dafsys S.R.L., Buenos Aires, 54.1334.1871
Australia, Email Electronics, Huntingdale, 011-61-3-544-8244; NSD-Australia, Victoria, 03 8900970
Brazil, Microlinear, Sao Paulo, 5511-220-2215
Chile, Sisteco, Santiago, 562-234-1644
China/Hong Kong, Novel Precision Machinery Co., Ltd., Kowloon, Hong Kong, (852) 360-8999
Guatemala, Abinitio, Guatemala City, 5022-32-4104
India, Priya International Limited, Bangalore, 91-812-214027, 812-214395; Priya International Limited, Bombay, 91-22-2863611, 22-2863676, 22-2863900, 22-2864026; Priya International Limited, New Delhi, 91-11-3314512, 11-3310413; Priya International Limited, Madras, 91-44-451031, 44-451597; Priya International Limited, Secunderabad, 91-842-813549, 842-813120; SES Computers and Technologies Pvt. Ltd., Delhi, 91-11-6849285, 11-6843006; SES Computers and Technologies Pvt. Ltd., Bombay, 91-22-4939977, 22-4943731; SES Computers and Technologies Pvt. Ltd., Bangalore, 91-812-348481, 812-343685
Jamaica, MC Systems, Kingston, (809) 926-0104
Japan, Asahi Electronics Co. Ltd., Kitakyushu-shi, 093-511-6471; CTC Components Systems Co., Ltd., Kanagawa 213, 044-852-5121; Dia Semicon Systems, Inc., Tokyo 154, 03-3439-1600; Okaya Koki, Nagoya-shi, 052-204-8315; Ryoyo Electro Corp., Tokyo 104, 03-3546-5011
Korea, J-Tek Corp., Seoul, (822) 557-8039; Samsung Electronics, Seoul, (822) 751-3680
Mexico, PSI S.A. de C.V., Cuernavaca-MOR, 52-73-11-1994/5
New Zealand, Email Electronics, Penrose, Auckland, 011-64-9-591-155
Saudi Arabia, AAE Systems, Inc., Sunnyvale, CA U.S.A., (408) 732-1710
Singapore, Electronic Resources Pte, Ltd., (65) 283-0888
South Africa, Electronic Building Elements, Pretoria, 011-2712-803-7680
Taiwan, Micro Electronics Corp., Taipei, R.O.C., (886) 2-7198419; Acer Sertek, Inc., Taipei, R.O.C., (886) 2-501-0055
Uruguay, Interfase, Montevideo, 5982-49-4600
Venezuela, Unixel C.A., Caracas, 582-238-7749

*Technical Distributor
†VAD